2. The method of claim 1, wherein the programming objects have interfaces through which the methods can be accessed.

3. The method of claim 1, wherein the programming objects comprise COM objects.

4. The method of claim 1, wherein said factoring comprises creating a hierarchy of object interfaces in which certain interfaces can inherit from other interfaces.

5. The method of claim 1, wherein said factoring comprises creating a hierarchy of object interfaces in which certain interfaces can aggregate with other interfaces.

6. The method of claim 1 further comprising instantiating a plurality of programming objects across a process boundary.

7. The method of claim 1, further comprising instantiating a plurality of programming objects across a machine boundary.

8. The method of claim 1, wherein the criteria is based, at least in part, on the manner in which particular functions behave.

9. The method of claim 8, wherein the manner includes a consideration of the types of operating system resources that are associated with the operation of a function.

10. The method of claim 8, wherein the manner includes a consideration of whether a particular function creates an operating system resource.

11. The method of claim 8, wherein the manner includes a consideration of whether a particular function operates upon an operating system resource.

12. The method of claim 1, wherein the criteria is based, at least in part, on the manner in which particular functions behave, wherein the manner includes:

a consideration of the types of operating system resources that are associated with the operation of a function; and

a consideration of whether a particular function creates an operating system resource.

13. The method of claim 1, wherein the criteria is based, at least in part, on the manner in which particular functions behave, wherein the manner includes:

a consideration of the types of operating system resources that are associated with the operation of a function call; and

a consideration of whether a particular function operates upon a given operating system resource.

14. (Amended) A method of factoring operating system functions comprising:

factoring a plurality of operating system functions that are used in connection with operating system resources into first groups based upon first criteria;

factoring the first groups into individual sub-groups based upon second criteria; and

assigning each sub-group to its own programming object interface, wherein a programming object interface represents a particular object's implementation of its collective methods effective to provide an object-oriented operating system.

15. The method of claim 14, wherein the first criteria is based upon the type of resource that is associated with an operation of a function.

16. The method of claim 14, wherein the second criteria is based upon the nature of an operation of a function on a particular resource.

17. The method of claim 16, wherein said nature concerns whether a function creates a resource.

18. The method of claim 16, wherein said nature concerns whether a function does not create a resource.

19. The method of claim 14, wherein the first criteria is based upon the type of resource that is associated with an operation of a function, and the second

criteria is based upon the nature of an operation of a function on a particular resource.

20. The method of claim 14, wherein at least one interface inherits from another interface.

21. The method of claim 14, wherein at least one interface aggregates with another interface.

22. The method of claim 14 further comprising instantiating a plurality of programming objects across a process boundary.

23. The method of claim 14 further comprising instantiating a plurality of programming objects across a process boundary and a machine boundary.

24. (Amended) A method of factoring operating system functions comprising:

factoring a plurality of operating system functions into interface groups based upon the resources with which a function is associated;

factoring the interface groups into interface sub-groups based upon each function's use of a handle that represents a resource; and

organizing the interface sub-groups so that at least one of the interface sub-groups inherits from at least one other of the interface sub-groups.

25. The method of claim 24, wherein said organizing comprises aggregating at least one of the interface sub-groups.

26. The method of claim 24, wherein the interface sub-groups are associated with COM objects.

27. The method of claim 24, wherein the factoring of the interface groups into interface sub-groups comprises considering whether a function creates a handle.

28. The method of claim 24, wherein said organizing comprises aggregating at least one of the interface sub-groups, and wherein the factoring of the interface groups into interface sub-groups comprises considering whether a function call creates a handle.

29. (Amended) An operating system application program interface embodied on a computer-readable medium comprising a plurality of object interfaces, wherein each object interface is associated with an object that includes one or more methods that are associated with and can call functions of an operating system that does not comprise the object interfaces, individual objects being configured to be instantiated in process, locally, or remotely.

30. The operating system application program interface of claim 29, wherein the object interfaces are arranged in groups in accordance with the types of objects with which their operation is associated.

31. The operating system application program interface of claim 29, wherein the methods within some of the interfaces are arranged in accordance with whether they create an object.

32. The operating system application program interface of claim 29, wherein the methods within some of the interfaces are arranged in accordance with whether they do not create an object.

33. The operating system application program interface of claim 29, wherein the methods within some of the interfaces are arranged in accordance with whether they operate upon an object.

34. The operating system application program interface of claim 29, wherein at least some of the object interfaces are arranged so that they inherit from other of the object interfaces.

35. The operating system application program interface of claim 29, wherein at least some of the object interfaces are arranged so that they aggregate with other of the object interfaces.

36. An operating system comprising:

a plurality of programming objects having interfaces, wherein the programming objects represent operating system resources, and wherein the

interfaces define methods that are organized in accordance with whether they create an operating system resource or not;

wherein the programming objects are configured to be called either directly or indirectly by an application; and

wherein the methods are configured to call operating system functions responsive to being called directly or indirectly by an application.

37. The operating system of claim 36, wherein some of the objects are disposed across at least one process boundary.

38. The operating system claim 36, wherein some of the objects are disposed across at least one machine boundary.

39. (Amended) The operating system of claim 36, wherein at least some of the objects are disposed across at least one process boundary and at least one machine boundary.

40. (Amended) The operating system of claim 36, wherein the objects comprise COM objects.

41. (Amended) A method of converting an operating system from a non-object-oriented format to an object oriented format, wherein the operating system includes a plurality of operating system functions that are callable to create or use operating system resources, the method comprising:

defining a plurality of programming object interfaces that define methods that correspond to the operating system functions, wherein programming objects that support the interfaces are callable either directly by an application that makes object-oriented calls, or indirectly by an application that makes function calls;

calling a programming object interface either directly via an object-oriented call, or indirectly via an indirection that transforms a function call into an object-oriented call; and

responsive to said calling, calling an operating system function with a method of the programming object that supports said programming object interface.